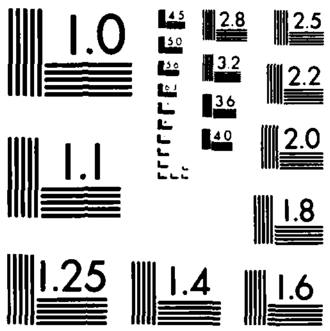
AD-A149 395 HUMAN FACTORS IN COMMAND CONTROL AND COMMUNICATION SYSTEMS(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA R S NICKERSON OCT 79 BBN-4236 MDA903-76-C-0207 1/1

UNCLASSIFIED F/G 5/5 NL

END
FILMED
DTIC


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Bolt Beranek and Newman Inc.

[illegible]

Report No. 4236

Human Factors in Command, Control and Communication Systems

October 1979

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 4236	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Human Factors in Command, Control and Communication Systems		5. TYPE OF REPORT & PERIOD COVERED Final Report Feb.1976-Sept. 1979
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. S. Nickerson		8. CONTRACT OR GRANT NUMBER(s) MDA903-76-C-0207
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE October 1979
		13. NUMBER OF PAGES 4
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Unlimited distribution		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Command, Control and Communication; Human Factors; Psychology; Displays; Information Processing; Human Performance; Memory; Attention; Decision Making; Problem Solving; Stress; Interactive Systems; Person-Computer Interaction		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This is the Final Report for Contract MDA903-76-C-0207. Work done under the contract is described in a set of technical reports listed herein.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT NO. 4236

HUMAN FACTORS IN COMMAND, CONTROL AND COMMUNICATION SYSTEMS

OCTOBER 1979

PREPARED FOR:

Defense Advanced Research Projects Agency

Report No. 4236 Bolt Beranek and Newman Inc.

Human Factors in Command, Control and Communications Systems

R. S. Nickerson

Final Report

Contract No. MDA903-76-C-0207

October 1979

Submitted to:

Defense Advanced Research Projects Agency

1400 Wilson Boulevard

Arlington, VA 22209

This is the final report on Contract No. MDA903-76-C-0207, Human Factors in Command, Control and Communications Systems. The initial purposes of this contract were to review research on information processing by humans and relate the results to C^3 systems and operations, to survey the state of the art of man computer interaction, to organize a series of meetings on topics of relevance to human factors of C^3 systems, and to develop a plan for a program of research pertaining to human factors problems in C^3 systems.

The contract has been amended twice. These Amendments added several tasks relating to studying the feasibility of implementing a memory augmenter that would allow retrieval from a structured data base using the types of cues that are thought to be effective as retrieval aids in human memory, and also the task of assisting in the editing of a special issue of _Human Factors_ on Human Factors and Computers and preparing an article based on work performed under the contract for that issue.

Work performed under the original work statement was described in a two-volume report submitted to ARPA in February 1977. In keeping with one of the objectives of the amendments a report was submitted to ARPA on memory augmentation in July 1978.

Work on the preparation of the special issue of _Human Factors_ on Human Factors of Computer Systems has proceeded in collaboration with personnel at the Army Research Institute. Invitations were sent to several appropriate researchers to contribute papers to the special issue. In addition, an announcement was published in the Bulletin of the Human Factors Society issuing a general call for such papers. The response was generally disappointing in terms both of quantity and quality of submissions. Nevertheless, a sufficient number of acceptable papers has now been received to justify a special issue. Also some other potential contributors have agreed to participate. Appended to this report is the paper that was prepared under this contract for the special issue.

During the course of this project several technical papers were prepared describing work performed under the contract. These reports are listed in Table 1.

Table 1. Reports prepared under Contract No. MDA903-76-C-0207.

Nickerson, R. S. Some characteristics of conversations. _Proceedings of NATO Advanced Study Institute Conference on Man-Computer Interaction_, Mati, Greece, 1976 (in press). Also, BBN Report No. 3498.

Nickerson, R. S. On conversational interaction with computers. In S. Treu (Ed.), _User-oriented design of interactive graphics systems, Proceedings of ACM/SIGGRAPH Workshop, 14-15 October 1976_, Pittsburgh, PA, 101-113. Also, BBN Report No. 3499.

Nickerson, R. S., Adams, M. J., Pew, R. W., Swets, J. A., Fidell, S. A., Feehrer, C. E., Yntema, D. B., & Green, D. M. The C -system user. Vol. 1: A review of research on human performance as it relates to the design and operation of command, control and communication systems. Vol. II: Workshop Notes. BBN Report No. 3459 (submitted to Defense Advanced Research Projects Agency, February 1977).

Nickerson, R. S. Some comments on human archival memory as a very large data base. _Proceedings on Very Large Data Bases_, Third International Conference on Very Large Data Bases, Tokyo, Japan, 6-8 October 1977, 159-168.

Nickerson, R. S., & Adams, M. J. Long term memory for a common object. _Cognitive Psychology_, 1979, _11_, 287-307.

Nickerson, R. S. Why interactive computer systems are sometimes not used by people who might benefit from them. Prepared for a special issue of _Human Factors_ on Human Factors and Computers.

APPENDIX

Why Interactive Computer Systems are Sometimes Not Used by People Who Might Benefit from Them

R. S. Nickerson

Why Interactive Computer Systems are Sometimes Not Used by People Who Might Benefit from Them

R. S. Nickerson

ABSTRACT

Several reasons are considered why some people who might benefit from using computer systems do not use them. The discussion is organized around examples of several classes of complaints that abstainers and dissatisfied users have been known to make regarding various aspects of the design and operation of specific computer-based systems.

A large number of computer systems have been developed for one or another community of users. Many of these systems are not used by the people for whom they were intended. The purpose of this paper is to consider some of the reasons why such systems--and especially those that appear to have considerable potential for their intended users--are not used.

The observations that are made derive primarily from informal interviews and conversations with users and potential users of a variety of systems. Complaints that are reported include those registered by users who cont to work with systems in spite of what they perceive to their shortcomings, as well as those offered by non-users as ons for their abstinence. The literature is cited in a few instances, but no effort has been made to conduct a thorough search of it. Speculation is not avoided. All in all, the ideas that are presented are best viewed as conjectures or hypotheses. Some of them may be intuitively compelling to computer users, or would-be users; in other cases there is undoubtedly a need for empirical verification.

Before turning to the main theme of the paper, a comment is in order about its focus. Some computer systems are used because their users have no choice in the matter. That is to say the users must use the systems or they cannot perform their jobs. Examples include airline reservation systems, hotel reservation systems, and sales registration systems. My interest in this paper is in systems that do not have to be used and in users, or potential users, who are not required to be users by virtue of their jobs. This delimitation of the problem will have the effect of focussing attention primarily on people, such as managers and professionals, who tend to have considerable latitude in how they perform their jobs, as opposed to technicians and clerical personnel whose tasks are likely to be more tightly prescribed.

In what follows, a variety of "gripes" that are representative of those expressed by users and potential users of interactive systems are considered under each of several topics. The organization is somewhat arbitrary and has been adopted for the purpose of facilitating the presentation; there is no intention to suggest that this is the only, or even necessarily the best, way of structuring the problem area. Each section begins with a sample of related gripes, which, hopefully, motivates the discussion that follows.

Functionality

"The system is very impressive, but it doesn't offer me anything I really need."

"I find I can do my job better and with less frustration if I do it manually than if I try to use the system."

"The system clearly was not designed with my job in mind."

It is undoubtedly true that some systems have been developed to meet needs that existed only in the minds of their developers. The failure of the target users to use such systems should not be surprising.

Lest this observation be misinterpreted, we should note also that although there are good reasons for involving potential users of a system in its initial design, such involvement does not guarantee that the eventuating system will be used. The assumption that the intended user knows what he needs and can convey that knowledge to the system developer is probably not always a safe one to make. A manager, for example, may not know, or be able to say explicitly, how he does his job; and even if he can describe what he does, he may be unable to say what he needs by way of help from a computer. Also, a job may change with the introduction of new tools. While a system developer should seek the advice and counsel of the intended user of the system that he expects to develop, he would do well to stay alert to the possibility that there may prove to be a difference between the problem as stated and the real problem that he has to solve. If he fails to maintain sensitivity to this possibility, he may discover that he has succeeded in building a system that is not used in spite of the fact that it was built to user specifications.

While acknowledging that some systems are not used because they do not address real needs, I believe that many systems that are not used would facilitate the intended user's performance of his job if he took advantage of their existence. The question of interest from a human factors point of view, and the question that is addressed in the remainder of this paper is why such potentially useful systems are not used.

Accessibility-Availability

"I share a terminal with several other users, so often have to wait for it."

"Frequently when I try to dial in, all the lines are busy, so I have to wait. What is particularly frustrating is not knowing how long I should wait before trying again."

"The computer is often down when I want to use it."

Ideally, the user would like to have immediate access to the computer on a continuing basis. He would like to be able to work with it on a problem whenever, and for however long as, he has the inclination to do so. This may be a difficult objective for many systems, for the present at least, because it requires that every user have a private terminal devoted exclusively to his own use, that a single system not have more subscribers than it can accommodate simultaneously, and that the system be backed up by an alternative for coverage in the event of system failure.

But if it is clear that guaranteeing immediate access to all users on a continuing basis is a difficult objective, it is equally clear that it is possible so to saturate a system and so to limit access to it as to make some potential users reject it out of hand. One of the most common operating objectives of time-sharing systems is that of keeping the central processor maximally busy. One way to be certain to do this, of course, is to structure the situation so that there is always a queue of users waiting to get on line. One can only justify this policy by placing a very low value on the user's time--which seems to be at least as contrary to the purpose of time sharing as is the wasting of computer time. Clearly, some compromise is to be preferred to either extreme.

What is not clear is what constitutes acceptable access from the user's point of view. Is it better to have guaranteed access during limited periods of time, or to have limited access (e.g., first-come first-served until saturated) all the time? Given a first-come first-served system, what will the user consider to be an acceptable probability of obtaining access when he wants it? How frequently will he tolerate being turned away before he gives up on the system altogether? Or, given a priority scheduling scheme, how does his position on the priority scale affect his interaction with the system? How frequently will one tolerate being pre-empted by a higher-priority user?

These and similar questions point up the fact that in determining what sorts of trade-offs are reasonable, closer attention will have to be given to user preferences, and to the effects of different scheduling policies and their attendant access implications on the successfulness of the person-computer interaction.

Start-stop Hassle

"It's too much trouble getting started. I have to dial a phone number, get on a network, identify the computer I want to use, login on it, call up the software system with which I want to work, and retrieve my files before doing anything productive."

"Exiting from a work session is not much easier than getting started. It should be possible just to say 'goodbye' and leave."

Licklider (1960), in his classic paper on man-computer symbiosis, observed that people engaged in intellectually demanding work can spend a surprisingly large fraction of their time "getting into a position" to think. When one is using computer-based tools to facilitate the accomplishment of intellectually demanding tasks, this positioning time includes time that is required for the user to access the particular software system with which he wants to interact and to initialize it for the needs of that particular work session. Sometimes the preliminaries are frustratingly involved.

Ideally, the user would like the computer to take care of as many as possible of these preliminaries automatically. In essence, what one wants is the ability to say to the computer "It's me" and have it do all the appropriate things to deliver the software tools that one wants. This is, of course, feasible only for users who tend to want the same tools on different occasions. Such users are not unusual, however, and even for those who use an assortment of tools, a default selection of the most commonly requested one could be a significant help.

A particularly helpful capability for a system to have is one that would permit the user to return to the computer after the termination of a work session, announce his return with something like "I'm back," and have the system automatically reestablish the situation to exactly what it was when he left. To use the analogy of a conventional workspace, what one wants to be able to do is leave the work situation for an indefinite time, and upon returning, find everything as it was left when one went away. Most systems make it possible for the user to restore things more or less as they were when a work session was terminated, but the burden is on the user to make sure he leaves things in a restorable state (e.g., that he files the material on which he is working), and to take certain explicit steps to effect the restoration.

Terminating a work session is also sometimes more cumbersome than one would like it to be. One should not have to disconnect, explicitly, every connection that was established in getting the session underway. If, for example, in order to gain access to a particular applications program the user had to make connections with several layers of systems and subsystems, he should not have to break each of those connections explicitly in terminating the session; a single sign-off command should suffice.

One of the consequences of the fact that work session initiation or termination is something of a hassle, or at least is perceived to be, is that users sometimes stay logged on a system for long stretches of time during which they are not actually using it, simply to avoid the necessity of terminating and restarting.

System Dynamics and Response Time

"The system is too slow."

"What's worse than waiting for a response is not knowing what the duration of the wait is going to be."

"Sometimes when I am waiting for a response, I don't know whether the system is alive or dead. If I knew it was dead, I could go do something else."

There seems to be general agreement among investigators of person-computer interaction that long delays in system response can be frustrating, and can decrease the efficiency of the work session. There is not general agreement, however, regarding what constitutes a long delay, or regarding the conditions under which such a delay will have a detrimental effect.

Elsewhere (Nickerson 1969), I have suggested that three factors may play significant roles in determining whether a given delay will be objectionable:

a. The user's _uncertainty_ regarding the duration of the delay.

b. Failure of the system to meet the user's _expectations_.

c. The perceived <u>cause</u> of the delay.

A long delay may be more tolerable if one can predict when it will end than if one cannot. A convention that is sometimes used is that of having the system tell the user if there is to be an unusually long delay, if it can (e.g., when it has to retrieve a file). The user's expectations may be disappointed in two ways: When the system appears to be more sluggish than one expects it to be on the average (a characteristic that may vary with the load on the system), and when it is unexpectedly slow in responding to a particular user input. That the acceptability of a delay may depend on what is perceived to be the cause of it seems clear. One would expect the user to be more tolerant of a long delay, for example, if he believes the computer is working on his program than if he thinks it is ignoring him.

Miller (1968) has suggested that whether a delay will be frustrating will depend on whether it follows a point of closure within the work session or comes during the process of obtaining closure. The assumption is that it is more important to maintain the continuity of a thought process at some phases of problem solving than at others and that an unnecessary delay will be particularly disruptive if it comes at a time when continuity should be preserved. Whether a particular delay will be disruptive also may depend somewhat on the level of expertise of the user and on the purpose for which the system is used.

The notion that the relationship between efficiency of person-computer interaction and system response time may be discontinuous has been expressed by several writers (Carbonell, Elkind, & Nickerson, 1968; Miller, 1968; Simon, 1966). Miller suggests that ideally delays should not exceed about 2 seconds, and that delays of 15 seconds or longer rule out conversation-like interactions. Simon suggests the possibility of quantizing the system's response time; the idea being either to make the response immediate or, if an appreciable delay is necessary, to force it to be of fixed and known duration (say, several minutes). In general, how important it is to maintain short delays probably depends on how important it is that the interaction have a conversation-like character.

One solution to the response-time problem, but not necessarily an economically feasible one, is that of keeping a system sufficiently lightly loaded to provide all users with the capacity necessary to preclude delays. A more attractive approach economically is to reserve some proportion of a system's resources for the processing of "background" jobs that do not require user interaction, and to adjust this proportion continuously so that it always complements the load represented by the demands of interactive use. Still another solution, and one which will probably be used increasingly in the future, is to provide each user with sufficient local capacity--resident in his terminal--to make reliance on the central computer for word processing and display generation functions unnecessary.

Two particularly important tasks for a vendor of time-shared computing resources are: (1) making sure that buyers have an accurate understanding of the kind of responsiveness they can expect, and how their behavior can affect that responsiveness, and (2) making available, on demand, some indication of what is being delivered to the user at any given time. Time-shared resources are often sold on an "at-least" basis. That is to say, the buyer is guaranteed that for a certain price the system will deliver to him a certain fraction of its total resources, at least. However, in such cases a system may often deliver more than the guaranteed minimum. If, for example, a system is lightly loaded (undersold) it might deliver 20% of its total resources to a customer who had paid for a guaranteed 10%. This sounds like a good deal from the buyer's point of view. The problem is he gets used to the idea of getting the windfall, comes to assume that that is what he should expect, and begins to rely on it. Things are fine until the excess capacity that was going into windfall is sold to new customers and therefore disappears. The user who was getting the windfall is now angry because something he had come to rely on, and to perceive as rightfully his, has been taken away. The system that was once snappy now appears to be very sluggish, because he is trying to get from 10% of the capacity what he had been getting from 20%.

One can make a fairly convincing argument that it is better never to let a user get used to windfall than to let him get used to it and then take it away from him. Pushed to the extreme this position can be used to support the notion that the excess capacity of an undersold system should be wasted rather than divided among the paying users. An alternative approach is that of providing for the user detailed information regarding what percentage of the system's resources are being delivered to him at any given time. When a user is benefiting from windfall, such information would serve as a constant reminder both of the fact and of the amount; when there is no windfall the information would assure the user that he is receiving exactly that percentage of the resources he had been guaranteed (assuming that he is). He would, of course, be free to load his portion of the machine so heavily as to cause the response time to be very long, but at least he would know that the system's behavior was due to the load he was imposing and not to the fact that he was not getting the percentage of the resources for which he was paying.

Work-session Interrupts

"The system frequently crashes when I am in the middle of a work session; and often considerable work is lost as a consequence."

"Usually system crashes occur without any warning."

"The user is not told, when a crash occurs, how long it will be before the system will be up again."

It is not surprising that interruptions of work sessions are both annoying and disruptive. If such interruptions occur too frequently or are too devastating when they do occur, users will simply forsake the system and find other ways to get their work done. Or, if they are truly captive of the system because they cannot do their work without it, they will become exceedingly unhappy.

The problem of system crashes is a challenge both to system designers and to researchers interested in the study of person-computer interaction. The designer's problem is not only that of minimizing the frequency and duration of crashes but that of making the effect of crashes as painless as possible. Providing warnings that a crash is about to occur is helpful, if it can be done. Minimizing the negative impact of a crash, in terms of lost work, by backing up files is also helpful. And providing the user with some indication of how long the system is likely to be down, if possible, relieves the considerable annoyance that may be attributed simply to uncertainty, and the consequential inability to plan effectively the use of one's time.

A challenge to the researcher who is interested in person-computer interaction is to determine the behavioral effects of work session interrupts of different types and to quantify their implications for user attitudes and performance. Any unanticipated interruption is likely to be annoying; however, how disruptive it is, and how much it impairs the user's performance may depend on the specifics of the situation. It would be helpful to have a better understanding of those dependencies.

Training and User Aids

"I don't have the time to learn how to use the system."

"Effective use of the system depends on knowing too many details."

"When I ask the system for 'help' it gives me messages that are not really helpful. Sometimes they tell me things I already know; sometimes they give me information that I cannot understand."

"The only effective way I've found to learn how to do something new with the system is to get someone who already knows how to do it to show me. Often I don't ask because its an imposition on one's time."

There are many systems in existence that are easy to use <u>provided</u> the user has the necessary experience with, and understanding of, them. Typically, however, such knowledge and experience are gained only at a considerable cost in terms of time, effort and perhaps frustration on the part of the user. To the expert, the system may be easy to use but only because he has long since mastered a myriad of details (of the sort that are often not considered to be important enough to document) that are necessary to make a smooth interaction possible. The system that can produce an instant expert user or make a novice user behave as though he were an expert has yet to emerge.

It may be unreasonable to expect such a system ever to be developed, although we may foster visions of such a thing when we talk about "easy to use" systems. It may be in the nature of things that any system that will aid human cognition in nontrivial ways will (at least for the immediate future) require a substantial investment of time for learning how to use it effectively. If this is true, then the problem is to provide to the potential user the information he needs to determine how much of an investment he must make and what advantages expertise with the system will provide him when and if he acquires it. The disillusionment that many would-be users experience may well be a result of discovering that the difficulty of learning to use a system effectively is greater than they had been led to believe.

Two types of training material are desirable for any person-computer system: that which will introduce one to the system and that which will facilitate the advancement of a user from novice to expert status. Potential users of a system are likely to be much concerned about how great an investment of their own time and energy they must make in order to acquire whatever degree of expertise is necessary to use a system to advantage on their own substantive problems. A fence straddler becomes a convert when he is successful in solving some problem with the help of a computer more effectively than he could have done so without it. The developer of a user-oriented system who wants the system to be used would do well to give some careful thought to how to structure things so the user can do something of <u>genuine interest</u> very early on. It is not reasonable to expect, of course, that a novice user should be able to exercise the full power of a system the first time he uses it, but he should be able to do something he perceives as nontrivial and genuinely helpful, and that will provide the reinforcement to learn more.

Introductory training material should be designed, therefore, to bring the beginning user to the point of accomplishing something of interest quickly. The further training that is necessary to increase the user's capability with the system should be provided both by conventional documentation and by training facilities incorporated within the system itself. Ideally, the training facilities incorporated within the system should include the ability to monitor a user's skill level and to volunteer information at opportune times in order to encourage and facilitate the acquisition of new knowledge and user skills.

The last point deserves emphasis. Many systems have the ability to provide help to the user on demand, but few have the ability to make suggestions in the absence of user-initiated requests. Such a capability is important for training purposes, inasmuch as a beginning user not only lacks knowledge about how to use the system effectively, but he also does not know what it is he does not know, and therefore, often does not know how to ask for information effectively. Moreover, some evidence has been obtained that suggests that if the user of a system can accomplish an objective in one way, he is unlikely to invest the time required to learn how to do it a second way even if the second way is the more efficient of the two (Eason, 1976). To the extent that this is true, it establishes the need not only for training material that will provide the user who is motivated to acquire more effective skills an opportunity to do so, but that will also help motivate other users to acquire those skills.

Before leaving the subject of training, it is probably worth noting that the needs in this regard may differ considerably among different classes of users. Some investigators, Eason (1976) and Stewart (1976), for example, have argued that the real challenge to designers of user-oriented systems is the manager user. As a rule, specialists tend to be willing to devote time and effort to the learning of how to use computer-based tools, providing the tools represent genuinely better ways of solving the problems with which they have to deal. Managers, on the other hand, are less likely to be willing to invest the necessary time and effort. Moreover, they are likely to be more demanding of evidence that a system, once learned, will really help them make more efficient use of their time and to solve the problems with which they have to deal more effectively than they otherwise could.

Documentation

"The documentation is ________."

The blank in the above statement may be filled in with any of a number of uncomplimentary adjectives: unclear, inaccurate, incomplete, poorly organized, out of date. Of course, documentation should not be any of these things; it should be clear, accurate, complete, well organized and current. So much is obvious. There are, however, several observations that can be made regarding desirable features of system documentation that are perhaps less apparent than these.

Documentation must serve both tutorial and reference functions. The beginning user needs to be able to read something that will introduce concepts and information in a logical sequence, and that will answer questions he does not know enough to ask. The more experienced user needs an information source (manual or computer file) to which he can go to find answers to specific questions. This is not to suggest that one source will not suffice for all users; it is simply an assertion of the importance of the distinction between tutorial and reference needs. If one source is to suffice for all users, it must be organized with this distinction in mind.

There is probably a need for both hard-copy and on-line documentation. Users often need assistance during the course of a work session, and in many cases what is needed could be provided on line. Many systems permit the user to ask certain types of questions regarding interactive procedures, and some are able to response to a general "help" command. How helpful the response is likely to be may depend upon a number of factors, such as the specifics of the situation in which the command is used, and the degree of sophistication of the user. One particularly frustrating experience for a novice user is to discover that he needs help to make effective use of the help command.

Because information can be presented on line is not by itself sufficient justification for presenting it that way. Some information may be better or more effectively presented in other ways. And, for the present at least, it is often helpful to have documentation available in hard-copy even if it is available on line as well.

Command Languages

"The commands that I have to use in order to instruct the computer seem arbitrary. The names by which the actions are identified are not always descriptive of those actions and, therefore, they are difficult to learn and to remember."

"The need to be letter perfect in designating commands is frustrating. The system should be smart enough at least to figure out when I have misspelled something in most cases."

"The dialog that the command language permits is stilted and unnatural; it lacks the flexibility and fluency of interperson conversations."

Mann (1975) has suggested that a major reason why people who are not computer specialists have difficulties learning to use computer systems effectively is the command-language nature of the interface that most systems provide. "Commands," he argues, "are an extremely narrow, limiting subset of people's familiar range of expression" (p. 2). A person makes relatively little use of commands when explaining to another person how to perform a task. Therefore, Mann suggests, if computer systems are to be made truly responsive to the needs of the computer-naive user, interactive languages must provide the ability to do more than issue commands; they must permit such things as the descriptions of objects, processes and goals, the negotiations of meanings, and the use of analogies and comparisons to clarify ideas.

While the notion that a "command language" would be an overly restrictive--and undoubtedly objectionable--vehicle for interpersonal communication is intuitively compelling, it does not follow that it is necessarily too restrictive for many types of computer uses. Moreover, the development of human-like conversational capabilities for person-computer interaction--assuming they are desirable--must wait not only on further developments in computer techniques for natural language processing, but also on a better understanding of interperson communication. For purposes of this paper I shall assume that command languages are an appropriate vehicle for person-computer interaction at the present time, and are likely to continue to be for the near-term future at least.

Command languages differ greatly for different systems. Some of these differences are probably due in part to differences in system capabilities. Some of them, however, are not. One can find fairly large differences among command languages for systems that were designated for the same purpose.

Languages have been designed for the most part in accordance with the intuitions of their designers. There is no theory of command language design or even a set of well-established design guidelines. Some assertions can be made which seem intuitively compelling, but they do not have the force of empirical substantiation.

It seems to be widely believed that it would be a good thing if one could talk with a computer in unconstrained natural language as one does with another human being. There are really two aspects to this notion, and it is important to distinguish between them; one could have a natural language capability without speech, or speech without natural language. A system with which one could communicate in unconstrained English via typewriter, for example, would have a natural-language capability but not speech. Alternatively, one that would recognize and/or produce spoken utterances chosen from a restricted vocabulary and produced in accordance with a specially designed syntax would have a speech capability, but not natural language. This distinction often is not clearly drawn in discussions of speech and/or natural language as computer input.

The great advantage of natural language as computer input is the fact that most people know how to use it moderately well. Among the disadvantages is the fact that it can be--perhaps typically is--vague and imprecise. Computer languages, in contrast, are precise to a fault--perversely literal, one might say--and, as a consequence, impose a discipline on the computer user that person-to-person communication does not. Whether, on balance, this discipline is a good or bad thing is a debatable point.

For the immediate future, the question of the desirability of a full fledged natural language capability is moot, because it is not a realistic goal, at least for most systems. (Limited vocabulary voice input systems are another matter, and several are currently in use [Martin, 1976].) It clearly is possible, however, to produce systems with more of the desirable features of natural language than systems typically now have. A greater tolerance than systems currently have for errors of spelling and syntax is achievable, for example. The ability to recognize synonyms could also be added to many systems and it should be relatively easy to provide users with the ability to define their own abbreviations and conventions. There are other things that can be done to make command languages more natural short of providing the full capability of natural language understanding (Nickerson, 1976).

The assumption that the preferred mode of interaction with a computer is via natural language--or something that approximates it as closely as possible--has led some system developers to endow their systems with characteristics that are calculated to convey the impression of a natural-language capability when, in fact, one does not exist. Thus, for example, a system might be given the ability to refer to the user by name, to insert folksy prattle into its output, and to project a "friendly-and-clever-fellow" image in other ways. It is at least a plausible conjecture that such superficial hints at natural-language ability can have the effect of making the interaction less, rather than more, natural than it otherwise might be, especially when the user discovers the fact that the intelligence he had been led to expect from the computer is not there. A question that deserves some attention from human factors researchers is that of how the effectiveness of the interaction between a computer and its user depends on (a) the conceptualization that the user has of the computer's capabilities, and (b) the veridicality of that conceptualization.

Consistency and Integration

"I get confused among the languages and conventions of the various systems and subsystems that I have to use. It's not only the fact that they differ but that they differ in particularly confusing ways. A given control character, for example, may mean one thing in one system and something else in another. Or conversely, to accomplish precisely the same thing one may have to use different command sequences in different systems."

"There is a need for standardization across systems. I don't like to have to remember that to delete the last character I typed I have to strike one control key if I am interacting with Program A and a different one if I am interacting with Program B."

The problem of lack of consistency and integration results from the fact that most large software systems are collections of components that were developed by different people. In designing the command languages for these components, each of the developers has used his own intuitions. Empirically demonstrated or even generally acknowledged guidelines have not existed.

The problem also stems in part from the fact that in order for the user to get at a particular applications program within a system, he typically has to interact with several programs (e.g., a network control program, executive or monitor, the applications program, etc.), each of which has its own command language or conventions.

To many users the need to think in terms of systems and subsystems is a negative factor. They would prefer to deal with a single integrated system with one command language. This desire is not likely to be realized to any great degree in the near future, because systems are likely to continue to be composites of individually developed components. This is not to say that nothing can be done in the interest of consistency, however.

One approach that has sometimes been proposed as a means of addressing the problem of lack of consistency among systems is that of developing a program that can act as an intermediary between a user and the various systems and applications programs that he may want to use. The idea is that of providing the user with an agent that would deal with some of the inconsistencies, much as a translator or interpreter deals with language problems in interperson communication, and relieve the user of the burden of learning many details regarding the operation of individual systems. The notion of a user-agent program is considerably more general than that of translator-interpreter, and there is the possibility of developing programs that could facilitate a user's interaction with computing systems in many ways. But intermediary programs exact costs because they use computer resources. Only the user can decide whether the benefits justify those costs.

User Conceptualization of System

"I don't understand what's going on within the system, and that makes me uncomfortable."

"I don't trust it. Maybe if I understood it better, I would."

"If the system can -----, why can't it -----?"

A particularly interesting question relating to the problem of designing computer systems that are to be used by people who are not knowledgeable with respect to computers, is that of how to describe those systems and their capabilities. What kinds of models or metaphors should be used to give one a helpful representation of a system, of what it can and cannot do, and of how it does what it does?

In the absence of guidance from the designer regarding how the system should be conceptualized, the user will undoubtedly invent his own model in time. There is no assurance, however, that the model will be useful, or that it will not have to be revised drastically as he learns more about the system.

The importance of the issue of users' models has been stressed by Newman and Sproull (1979), but very little effort has yet been devoted to the development of guideines for the invention and use of such models. This is, I believe, a ripe area for research. It would be instructive to know how the conceptualizations of different users of the same complex system may differ from each other when those users are left to their own devices to develop them.

Miscellaneous Other Factors

There are undoubtedly other reasons why some people refuse to use some computer systems, among them some that would not always be willingly acknowledged. Examples of such reasons:

- General resistance to change
- The feeling that direct interaction with a computer system is beneath one's position or status
- General mistrust of computer systems
- Fear that introduction of a computer will have a dehumanizing effect on a job situation

- Unfounded assumptions about what knowledge is required to be an effective user

- The prospects of replacement of procedures that are familiar and comfortable with those that are new and strange

- The threat of obsolescence or devaluation of hard-won skills

Some of these impediments could probably be removed by simply providing the information necessary to correct a misperception. Others, however, relate to more deeply seated problems. The introduction of a computer into an ongoing operation may facilitate (or complicate) the performance of familiar tasks. It is important to recognize, however, that it also may change the nature of those tasks in qualitative ways. Moreover, and perhaps more importantly, it may change not only an individual's job, but his perception both of the job and of its value. Margulies (1976) has emphasized the importance of considering job-satisfaction issues in evaluating any person-computer system. He argues that the computer revolution offers an opportunity to reverse certain trends of the industrial revolution that tended to result in dehumanizing jobs. Whereas in many industrial systems people had become automatons or appendages to a machine, the prospect of automating menial and repetitive tasks contains the promise of freeing the human being from this role and making him a "supervisor, guide and partner" of the machines with which he works. Not all people whose jobs might be changed as a result of computerization will view the prospects from this positive light, however, and their concerns need to be acknowledged and faced.

Fox (1977a,b) has suggested that large time-shared systems, or the groups that run them, often constitute a threat simply by virtue of their size. They may interfere with established working patterns and may impose new patterns of their own. Their staff, because of its remoteness and the general purpose-character of its mandate, may not be well integrated with the users of the system. The expensiveness of the equipment may also be a source of resentment by staff, especially by those who make no use of that equipment. Moreover, because large systems tend to be optimized to give service over a range of acti ities, they tend not to satisfy anyone fully. In contrast, small computers, Fox suggests, can be identified with the individual groups to whose activities they are dedicated. They can be attuned to the needs of those particular groups and completely under the groups' control. They pose no threat either by virtue of their expense or the visibility of a large professional staff. Moreover, to gain the advantages of a centralized facility without some of its drawbacks, small computers can be connected together into networks to provide for resource sharing and certain economies of scale. Whether or not one agrees with Fox's conclusions regarding the psychological implications of the size of computer systems and of other characteristics that relate to size, his analysis provides some food for thought.

A problem that needs more study is how to introduce change nondisruptively into an ongoing operation. There is a variety of psychological issues relating to this problem, having to do with such things as the fact that innovative techniques sometimes require a period of adjustment, which can be more or less traumatic on the part of the people who must learn to use them. They have implications for organizational structures and for the way people relate to each other in management hierarchies. They sometimes represent threats to people who have acquired specialized knowledge over may years which suddenly may be useless, and therefore no longer the basis for qualifying an individual as an expert whose skills are highly valued by the organization.

The need for more attention to the problem of how to introduce innovative procedures to busy professionals may be illustrated by reference to medical applications of computers and another observation by Fox (1977a). One of the reasons why computer-based tools have not been more widely used by medical practitioners, Fox claims, is that there has not been a sufficiently sharp distinction made between remedial and innovative roles of the computing facilities that have been introduced to practitioners. The point, as I understand it, is that innovative uses of computers should be clearly designated as such. The practitioner who agrees to use a system on the assumption that it is going to help him do the work he usually does, but more efficiently, and then, to his surprise, finds himself involved in an experiment that requires significant changes in his modus operandi, may become quickly disillusioned and alienated from further use of computing tools.

There is one impediment to the use of computer systems that differs somewhat from the others that have been considered in this paper. One might call it the critical-mass problem. Some computer-based tools are effective only if all of the members of some functionally defined group use them. Airlines and hotel reservation systems are cases in point. One of the reasons these systems work as well as they do is the fact that it is impossible to make an airline reservation or a hotel reservation except through the computer-based reservation system. As a consequence of this fact, the computer can be assumed to have complete and up-to-date information at all times regarding the availability of seats or rooms, and therefore can permit decisions to be made on the spot regarding additional reservations. If reservations could be made independently of the system, then the system would loose much of its effectiveness.

An example of a computer-based system whose usefulness may be severely limited unless its users constitute a critical mass is a computer-based message system. Such a system can facilitate communication within an organization greatly, but its usefulness is greatly diminished if not all members of the organization (with whom one wants to communicate) use it. If, for example, one wants to send an announcement to everybody, and one has to resort to a conventional method (e.g., hardcopy memo) for some individuals, one might as well use the conventional method for everybody.

Summary

While the problems mentioned above do not exhaust the list of reasons why people do not use interactive computer systems, it does, in my view, account for a sizeable fraction of the complaints one hears from people who have decided against using systems that are available to them, or who have made an attempt to use them and then have given up. Some of these complaints are also heard from people who do use computer systems, despite what they consider to be serious deficiencies. The fact that a person complains does not, of course, mean that he considers a system to be worthless. The degree of a user's frustration with a system is likely to reflect the magnitude of the disparity he perceives between the type and quality of service he receives from the system and what he believes it should be able to deliver.

It is tempting to try to impose some prioritizing on this list. One may assume that in a statistical sense some factors represent a greater deterrent to computer use than do others. On the other hand, it is quite clear that all of these factors are important and that any one of them can be responsible for displeasure on the part of a user or a decision by a would-be user to leave a system alone. Moreover, it is also clear that not all users or would-be users would weight these factors the same way. For some people, for example, reliability is of the utmost importance; whereas response time is of little concern. Conversely, there are others who are willing to tolerate a certain amount of downtime, providing that when the system is up and running it gives a very snappy response. For some people, documentation, user aids, and a very simple command language are extremely important. Others are willing to live with a system that is suboptimal in many of these ways providing only that it offers the functionality they need.

System designers should, of course, strive to eliminate the bases for all the complaints mentioned above. That will not do away with complaints, however, even though it will mean better (more useful and useable) systems; users and would-be users will assuredly invent new ones. And hopefully attention to these new complaints that emerge will yield further system improvements. Presumably, that is at least one of the ways in which progress is made.

REFERENCES

Carbonell, J. R., Elkind, J. I., & Nickerson, R. S. On the psychological importance of time in a time sharing system. Human Factors, 1968, 10, 135-142.

Eason, K. D. A task-tool analysis of manager-computer interaction. Proceedings of a NATO Advanced Study Institute, Mati, Greece, September 1976 (in press).

Fox, J. Medical computing and the user. International Journal of Man-Machine Studies, 1977a, 9, 669-686.

Fox, J. Some observations on fuzzy diagnosis and medical computing. International Journal of Bio-Medical Computing, 1977b, 8(4), 269-275.

Licklider, J.C.R. Man-computer symbiosis. I.R.E. Trans. on Human Factors in Electronics, March 1960, 4-11.

Mann, W. C. Why things are so bad for the computer-naive user. Information Sciences Institute, ISI/RR 75-32, March 1975.

Margulies, F. Evaluating man-computer systems. Proceedings of a NATO Advanced Study Institute, Mati, Greece, September 1976 (in press).

Martin, T. B. Practical applications of voice input to machines. Proceedings of the IEEE, 1976, 64, 487-500.

Miller, R. B. Response time in man-computer conversational transactions. Proceedings of the Fall Joint Computer Conference, 1968, 267-277.

Newman, W. M., & Sproull, R. F. Principles of Interactive Computer Graphics. Second Edition. New York: McGraw-Hill, 1979.

Nickerson, R. S. Man-computer interaction: A challenge for human factors research. Ergonomics, 1969, 12, 501-517. Also in IEEE Trans. Man-Machine Sys., 1969, MMS-10, 164-180.

Nickerson, R. S. On conversational interaction with computers. In S. Treu (Ed.), User-oriented design of interactive graphics systems, Proceedings of ACM/SIGGRAPH Workshop, 14-15 October 1976, Pittsburgh, PA, 101-113. Also BBN Report No. 3499.

Simon, H. A. Reflections on time sharing from a user's point of view. Carnegie Institute of Technology, Computer Science Research Review, 1966, 43-51.

Stewart, T.F.M. The specialist user. Proceedings of a NATO Advanced Study Institute, Mati, Greece, September 1976 (in press).